بسم الله الرحمن الرحيم

لغة البرمجة جافا

Java Programming Language

الدرس الرابع

العمليات في جافا (الجزء الثالث)

التحكم في التنفيذ :

لغة جافا تملك جميع تعليمات التحكم الموجودة في لغة C و ++C .

التعليمات هي : if-else , while , do-while , for , switch , goto

جميع التعليمات الشرطية تختبر صحة أو خطأ الشرط لتحدد مسار التنفيذ . مثلا التعبير A==B ، في هذا التعبير استخدمنا العملية الشرطية (==) لاختبار تساوي قيمة A لقيمة B و هذا التعبير يرجع true أو false . جميع العمليات العلائقية التي وردت في الفقرات السابقة يمكن استخدامها لانشاء تعليمة شرطية

في لغة جافا لا يمكن استخدام الأرقام كـ قيم بوليانية .. بينما في ++C يمكن عمل ذلك ، حيث أن القيمة التي لا تساوي الصفر تمثل true و القيمة التي تساوي ٠ تمثل false .

و إذا أردنا استعمال قيمة عددية كقيمة بوليانية يجب أولا تحويلها إلى قيمة بوليانية و بعدها يمكن استخدامها .

تعليمة if-else :

لها الشكل العام التالي :

```
if(Boolean-expression)
      statement
```

أو :

```
if(Boolean-expression)
      statement
else
      statement
```

الشرط يجب أن يولد قيمة بوليانية . كلمة `statement` تعبر إما عن تعليمة واحدة أو عدة تعليمات

مثال:

```
public class IfElse {

      static int test(int testval, int target) {

            if(testval > target)
                  return +1;
            else if(testval < target)
                  return -1;
            else
                  return  0;
```

```
	}

	public static void main(String[] args) {

		System.out.println(test(10, 5));
		System.out.println(test(5, 10));
		System.out.println(test(5, 5));
	}
}
```

تعليمة return لها وظيفتين :

- **تحديد القيمة المعادة من الطريقة**
- **إعادة هذه القيمة مباشرة للتابع المستدعي**

الحلقات :

الحلقة while :

لها الشكل العام التالي :

```
while(Boolean-expression)
	statement
```

الشرط يقيم في كل مرة تنفذ فيها الحلقة

مثال :

```
public class WhileTest {

	public static void main(String[] args) {

		double r = 0;

		while(r < 0.99d) {
			r = Math.random();
			System.out.println(r);
		}
	}
}
```

الطريقة `random()` في الصف `Math` تولد أعداء حقيقية بين الـ ٠ و الـ ١ طبعا بما فيهم الـ ٠ و الـ ١

الحلقة `do-while` :

ءلها الشكل العام التالي :

```
do
      statement
while(Boolean-expression);
```

الفرق بين while و do-while هي أن التعليمات الموجودة في جسم التعليمة do-while ستنفذ مرة واحدة على الأقل

<u>الحلقة for :</u>

لها الشكل العام التالي :

```
for(initialization; Boolean-expression; step)
      statement
```

يمكن أن تكون تهيأة المتحولات و الشرط و الخطوة واحد منها أو كلها فراغ .

يتم اختبار الشرط قبل بدء التنفيذ الحلقة و من ثم يتم تنفيذ جسم الحلقة و في النهاية تنفذ تعليمة الخطوة .

<u>مثال :</u>

```
public class ListCharacters {

      public static void main(String[] args) {

            for( char c = 0; c < 128; c++)
                  if (c != 26 )
                        System.out.println("value: " + (int)c +
                                           " character: " + c);

      }

}
```

يمكن تعريف أكثر من متحول في تعليمة for .. مثلا :

```
for(int i = 0, j = 1;
        i < 10 && j != 11;
        i++, j++)
      /* body of for loop */;
```

<u>Break & continue:</u>

داخل أي حلقة يمكن أيضا التحكم بالتنفيذ باستخدام break و continue .

Break تؤدي إلى الخروج من الحلقة بدون تنفيذ التعليمات التي بعدها

Continue تؤدي إلى انهاء التنفيذ الحالي للحلقة و تعود إلى أول الحلقة و من ثم تبدأ التنفيذ التالي للحلقة .

مثال :

```
public class BreakAndContinue {

      public static void main(String[] args) {

            for(int i = 0; i < 100; i++) {

                  if(i == 74) break; // Out of for loop
                  if(i % 9 != 0) continue; // Next iteration
                  System.out.println(i);

            }

            int i = 0;

            while(true) {

                  i++;
                  int j = i * 27;
                  if(j == 1269) break; // Out of loop
                  if(i % 10 != 0) continue; // Top of loop
                  System.out.println(i);

            }
      }
}
```

استخدام اللافتات مع break و continue :

اللافتة عبارة عن معرف متبوع بـ (:)

```
label1:
outer-iteration {
      inner-iteration {
      //…
      break; // 1
      //…
      continue; // 2
      //…
      continue label1; // 3
      //…
      break label1; // 4
      }
}
```

في الحالة الأولى تعليمة break تقطع تنفيذ الحلقة الداخلية

في الحالة الثانية تعليمة continue تقطع تنفيذ الحلقة الداخلية و تعود لبدايتها

في الحالة الثالثة تعليمة continue تقطع تنفيذ الحلقة الخارجية و تعود لبدايتها

في الحالة الرابعة تقطع تعليمة break تنفيذ الحلقة الخارجية

مثال :

```
public class LabeledFor {

      public static void main(String[] args) {

            int i = 0;
            outer: // Can't have statements here
            for(; true ;) {

                  inner: // Can't have statements here
                  for(; i < 10; i++) {

                        prt("i = " + i);
                        if(i == 2) {
                              prt("continue");
                              continue;
                        }
                        if(i == 3) {
                              prt("break");
                              i++; // Otherwise i never
                              // gets incremented.
                              break;
                        }
                        if(i == 7) {
                              prt("continue outer");
                              i++; // Otherwise i never
                              // gets incremented.
                              continue outer;
                        }
                        if(i == 8) {
                              prt("break outer");
                              break outer;
                        }
                        for(int k = 0; k < 5; k++) {

                              if(k == 3) {
                                    prt("continue inner");
                                    continue inner;
                              }
                        }
```

```
            }
        }
    }

    static void prt(String s) {
        System.out.println(s);
    }
}
```

(يتم تنفيذ البرنامج و مناقشة النتائج ....)

تعليمة switch :

لها الشكل العام التالي :

```
switch(integral-selector) {

    case integral-value1 : statement; break;
    case integral-value2 : statement; break;
    case integral-value3 : statement; break;
    case integral-value4 : statement; break;
    case integral-value5 : statement; break;
    // ...
    default: statement;

}
```

Integral-selector عبارة عن تعبير يولد قيمة صحيحة ، تعليمة switch تختبر القيمة السابقة في حال وجود تطابق مع أحد الحالات يتم تنفيذ الكود المقابل للحالة الموافقة . في حال عدم وجود تطابق مع أي حالة يتم تنفيذ الجزء default .

نلاحظ أن كل كتلة case تنتهي بـ break و التي تسبب قطع تنفيذ تعليمة switch ..

الشكل السابق هو الشكل الأكثر شيوعا لتعليمة switch ..

يمكن الاستغناء عن تعليمة break و في حال عدم وجودها يتم تنفيذ كل الحالات من بعد الحالة التي المطابقة

تعليمة Switch تفيد في في تضمين خيارات متعددة لشيء ما و ذلك عوضا عن استخدام بنى if-else المتداخلة

مثال :

```
public class VowelsAndConsonants {

    public static void main(String[] args) {

        for(int i = 0; i < 100; i++) {
            char c = (char)(Math.random() * 26 + 'a');
```

```
                System.out.print(c + ": ");
                switch(c) {
                    case 'a':
                    case 'e':
                    case 'i':
                    case 'o':
                    case 'u':
                        System.out.println("vowel");
                        break;
                    case 'y':
                    case 'w':
                        System.out.println(
                        "Sometimes a vowel");
                        break;
                    default:
                        System.out.println("consonant");
                }
            }
        }
    }
```